A Minor Project Final Report on

Brain Tumor Detection using ML

Submitted in Partial Fulfillment of the Requirements for

The Degree of **BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING**

Under **Pokhara University**

Submitted by:

Puspha Raj Pandeya, 171229

Sandesh Paudel, 171238

Under the supervision of

Er. Dileep Kumar T

Date: 02-10-2021

Department of Computer Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari Lalitpur, Nepal

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards our supervisor **Er. Dileep Kumar T** for helping us to prepare this project. This project helped us in numerous ways to expand our understanding of the processes involved, cost, and materials required for the development of this software. The continuous suggestions and encouragement were the source of inspiration for us.

We are also thankful to our friends and colleagues who are apart from the group for their continuous support and encouragement. Finally, I wish to record my appreciation to all the people who directly or indirectly contributed their help during the project work.

Thank You.

ABSTRACT

Automated detection of brain tumors in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment. The traditional method for tumor detection in magnetic resonance brain images is human inspection. This method is impractical due to a large amount of data. Hence, trusted, and automatic classification schemes are essential to efficiently detect brain tumors. So, automated brain tumor detection methods are developed as they would save radiologists time and obtain a tested accuracy. MRI brain tumor detection is a complicated task due to the complexity and variance of tumors. In this project, we overcome the drawbacks of the traditional method using the Deep Learning algorithm. We use Convolutional Neural Network for image classification with an accuracy of 97.5%.

Keywords*: Brain Tumor, Convolutional Neural Network, Deep Learning,*

Image Classifier, Machine Learning, MRI(Magnetic Resonance Image)

Table of Contents

List of Figures

List of Tables

List of Abbreviation

CNN	Convolutional Neural Network
MRI	Magnetic Resonance Imaging
VBM	Voxel-Based Morphometry
PCA	Principle Component Analysis
RM	Random Forest
SVM	Support Vector Machine
DTI	Diffusion Tensor Imaging
API	Application Programming Interface
UI	User Interface
IDE	Integrated Development Environment
CB	Cerebellum

1. Introduction

A brain tumor is one of the most rigorous diseases in medical science. Effective and efficient analysis is always a key concern for the radiologist in the premature phase of tumor growth. Histological grading, based on a stereotactic biopsy test, is the gold standard and the convention for detecting the grade of a brain tumor. Tumor biopsy being challenging for brain tumor patients, techniques like Magnetic Resonance Imaging (MRI) have been extensively employed in diagnosing brain tumors. Therefore, the development of systems for the detection and prediction of the grade of tumors based on MRI data has become necessary. But at first sight of the imaging modality like in Magnetic Resonance Imaging (MRI), the proper visualization of the tumor cells and their differentiation with its nearby soft tissues is a somewhat difficult task. This may be due to the presence of low illumination in imaging modalities or its large presence of data or various complexity and variance of tumors-like unstructured shape, viable size, and unpredictable locations of the tumor.

Automated defect detection in medical imaging using machine learning has become the emergent field in several medical diagnostic applications. Its application in the detection of brain tumors in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment. Studies in the recent literature have also reported that automatic computerized detection and diagnosis of the disease, based on medical image analysis, could be a good alternative as it would save radiologists time and also obtain a tested accuracy. Furthermore, if computer algorithms can provide robust and quantitative measurements of tumor depiction, these automated measurements will greatly aid in the clinical management of brain tumors by freeing physicians from the burden of the manual depiction of tumors.

The machine learning-based approaches like Deep ConvNets in radiology and other medical science fields play an important role to diagnose the disease in a much simpler way as never done before and hence providing a feasible alternative to surgical biopsy for a brain tumor. In this project, we attempted at detecting the brain tumor using Convolution Neural Network (CNN) architecture.

1.1. Problem Statement

Being in the era of Science and Technology we are still unable to utilize technology in a proper and managed way, but it doesn’t mean that we are not capable of doing so. The main

objective of this project is to identify major areas of image processing in brain tumor detection.

Brain tumors are a heterogeneous group of the central nervous systems that arise within or adjacent to the brain. Moreover, the location of the tumor within the brain has a profound effect on the patient's symptoms, surgical therapeutic options, and the likelihood of obtaining a definitive diagnosis. The location of the tumor in the brain also significantly alters the risk of neurological toxicities that alter the patient's comfort zone. No early detection strategies are in use, even in individuals known to be at risk for specific types of brain tumors by their genetic makeup. Current histopathological classification systems, which are based on the tumor‘s presumed cell of origin, have been in place for nearly a century and were updated by the World Health Organization in 1999. Although satisfactory in many respects, they do not allow accurate prediction of tumor behavior in the individual patient, nor do they guide therapeutic decision-making as precisely as patients and physicians would hope and need.

1.2. Project Objectives

Some of the core objectives of this project are listed below:

- Study and understand the literature that is based on image processing.
- Identify major areas of image processing in medicine and in particular on brain tumor detection.
- Identify current issues in brain tumor detection.
- Propose future directions and improvements that can be made in this area.
- Critically analyze and review identification of brain tumors using machine learning

1.3. Significance of Study

Brain Tumor is a type of disease that affects brain tissue. The risk of brain tumors may increase by stress, family history, chemical exposure, exposure to radiation, etc. By using traditional methods, there can be many errors that make health conditions more critical instead of making them better. The study helps in the improvement of human life. Accurate detection of brain tumors can be the difference between life and death. We use a modern method where machines are more prioritized than human beings because of their accuracy and effectiveness which reduces the death rate and also reduces the cost of the treatment.

1.4. Scope and Limitations

1.4.1 Scope

Here the scope of the project is that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcomes. This suggestion is promising as data modeling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment that can help to significantly improve the quality of clinical decisions.

1.4.2 Limitations

Medical diagnosis is considered a significant yet intricate task that needs to be carried out precisely and efficiently. The automation of the same would be highly beneficial. Clinical decisions are often made based on the doctor's knowledge and experience rather than on the knowledge-rich data hidden in the database. This practice leads to unwanted biases, errors, and excessive medical costs which affect the quality of service provided to patients. Data mining has the potential to generate a knowledge-rich environment that can help to significantly improve the quality of clinical decisions.

2. Literature Review

Numbers of studies have been done that focus on the detection of brain tumors. The application of artificial intelligence and machine learning algorithms has gained much popularity in recent years due to improved accuracy and efficiency in making predictions. They have applied different techniques for diagnosis and achieved different probabilities for different methods.

Ailion, A. S., King, T. Z., Wang, L., Fox, M. E., Mao, H., Morris, R. M., & Crosson, B. (2016), The cerebellum (CB) is known for its role in supporting processing speed (PS) and cognitive efficiencies. The CB often sustains damage from treatment and resection in pediatric patients with posterior fossa tumors. Limited research suggests that CB atrophy may be associated with the radiation treatment experienced during childhood. The purpose of the study was to measure cerebellar atrophy to determine its neurobehavioral correlates. Brain magnetic resonance images were collected from 25 adult survivors of CB tumors and age- and gender-matched controls ($M_{age=}$ 24 years (*SD*=5), 52% female). The average age at diagnosis was 9 years (*SD*=5) and the average time since diagnosis was 15 years (*SD*=5). PS was measured by the Symbol Digit Modality Test. To quantify atrophy, an objective formula was developed based on prior literature, in which Atrophy=[(CB White+CB Gray Volume)/Intracranial Vault (ICV)]$_{controls}$-[(CB White+CB Gray+Lesion Size Volume)/ICV]$_{survivors}$. Regression analyses found that the interaction term (age at diagnosis*radiation) predicts CB atrophy; regression equations included the Neurological Predictor Scale, lesion size, atrophy, and the interaction term and accounted for 33% of the variance in oral PS and 48% of the variance in written PS. Both interactions suggest that individuals with smaller CB lesion size but a greater degree of CB atrophy had slower PS, whereas individuals with a larger CB lesion size and less CB atrophy were less affected. Conclusion: The results of the current study suggest that young age at diagnosis and radiation is associated with CB atrophy, which interacts with lesion size to impact both written and oral PS. (*JINS*, 2016, *23*, 1–11) [1]

Ashburner, J., & Friston, K. J. (2000), At its simplest, voxel-based morphometry (VBM) involves a voxel-wise comparison of the local concentration of gray matter between two groups of subjects. The procedure is relatively straightforward and involves spatially

normalizing high-resolution images from all the subjects in the study into the same stereotactic space. This is followed by segmenting the gray matter from the spatially normalized images and smoothing the gray-matter segments. Voxel-wise parametric statistical tests which compare the smoothed gray-matter images from the two groups are performed. Corrections for multiple comparisons are made using the theory of Gaussian random fields. This paper describes the steps involved in VBM, with particular emphasis on segmenting gray matter from MR images with nonuniformity artifacts. We provide evaluations of the assumptions that underpin the method, including the accuracy of the segmentation and the assumptions made about the statistical distribution of the data. [2]

Aukema, E. J., Caan, M. W., Oudhuis, N., Majoie, C. B., Vos, F. M., Reneman, L., et al. (2009), To determine whether childhood medulloblastoma and acute lymphoblastic leukemia (ALL) survivors have decreased white matter fractional anisotropy (WMFA) and whether WMFA is related to the speed of processing and motor speed. For this study, 17 patients (6 medulloblastomas, 5 ALL treated with high-dose methotrexate (MTX) (4 × 5 g/m2) and 6 with low-dose MTX (3 × 2 g/m2)) and 17 age-matched controls participated. On a 3.0-T magnetic resonance imaging (MRI) scanner, diffusion tensor imaging (DTI) was performed, and WMFA values were calculated, including specific regions of interest (ROIs), and correlated with the speed of processing and motor speed. Mean WMFA in the patient group, mean age 14 years (range 8.9 – 16.9), was decreased compared with the control group (p = 0.01), as well as WMFA in the right inferior frontal-occipital fasciculus (IFO) (p = 0.03) and the genu of the corpus callosum (GCC) (p = 0.01). Based on neurocognitive results, significant positive correlations were present between processing speed and WMFA in the splenium (SCC) (r = 0.53, p = 0.03) and the body of the corpus callosum (bCC) (r = 0.52, p = 0.03), whereas the right IFO WMFA was related to motor speed (r = 0.49, $p < 0.05$).White matter tracts, using a 3.0-T MRI scanner, show impairment in childhood cancer survivors, medulloblastoma survivors, and also those treated with high doses of MTX. In particular, white matter tracts in the SCC, BCC, and right IFO are positively correlated with speed of processing and motor speed. [3]

Brett, M., Leff, A. P., Rorden, C., & Ashburner, J. (2001), In studies of patients with focal brain lesions, it is often useful to co-register an image of the patient's brain to that of another subject or a standard template. We refer to this process as spatial normalization. Spatial normalization can improve the presentation and analysis of lesion location in neuropsychological studies; it can also allow other data, for example from functional

imaging, to be compared to data from other patients or normal controls. In functional imaging, the standard procedure for spatial normalization is to use an automated algorithm, which minimizes a measure of the difference between image and template, based on image intensity values. These algorithms usually optimize both linear (translations, rotations, zooms, and shears) and nonlinear transforms. In the presence of a focal lesion, automated algorithms attempt to reduce image mismatch between template and image at the site of the lesion. This can lead to significant inappropriate image distortion, especially when nonlinear transforms are used. One solution is to use cost-function masking—masking the areas used in the calculation of image difference—to exclude the area of the lesion BCC that the lesion does not bias the transformations. We introduce and evaluate this technique using normalizations of a selection of brains with focal lesions and normal brains with simulated lesions. Our results suggest that cost-function masking is superior to the standard approach to this problem, which is affine-only normalization; we propose that cost-function masking should be used routinely for normalizations of brains with focal lesions. [4]

3. Methodology

We have planned to work on the following methodologies for the application of knowledge, skills, tools, and techniques to a broad range of activities to meet the requirements of our project. This section presents detailed information about the software development process, algorithms in machine learning, and datasets used for our project. It is a web application where we train a machine learning model to predict if the patient has a brain tumor or not.

3.1. Software Development Lifecycle

For this project, we will use the Iterative Incremental Model. Firstly, we gather user requirements and increase the developer‘s comprehension of the research area. Then, carried out to model user requirements into detailed computer-based specifications. A system and software design stage, architectural design, database, and interface design were developed. After the implementation, each module was tested as a unit to ensure the system fulfills business and design requirements. Brain Tumor Detection System is developed using the iterative incremental model as it is easy to understand and after the competition of each iteration, we can again start from the beginning and solve the problem that occurs at a particular phase. The user provides feedback on the product for the planning stage of the next iteration cycle and the development team responds, often by solving the problem that arises in the respective phase. This incremental cycle can continue until the product is shipped.

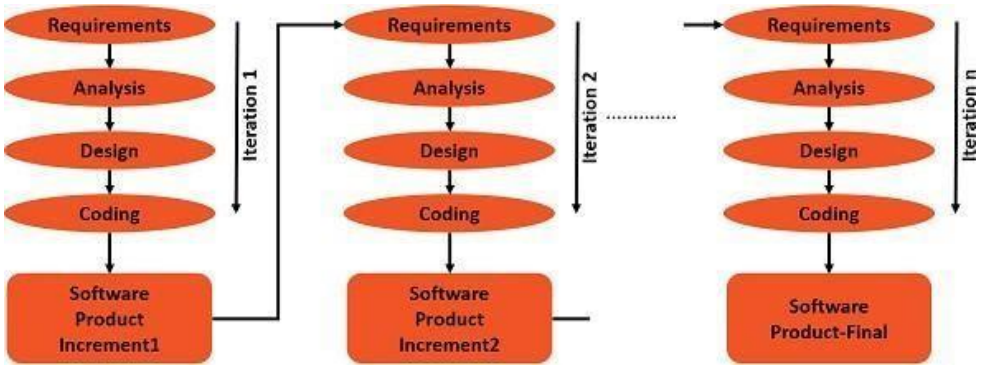

Figure 1 : Iterative Incremental Model

Iterative Incremental model includes the following phases:

Analytic Phase- In this phase, the requirements of the software were analyzed which resulted

in –**Software Required Specifications**‖.

Design Phase- In this phase, the analysis of SRS was translated into the system‘s design. Context Diagram, Use-Case Diagram, ER Diagram, and Class Diagram were developed.

Coding Phase- This phase involves the coding as per the design and formation of a working system at the end of the process.

Testing Phase- In this phase, the system was tested. With each testing, certain changes were made as per the suggestion. This was done incrementally until satisfaction was made.

In this project there are two iterations:

Iteration 1 includes the following activities

- Basic Design Diagrams
- Model building
- Creating Web API
- Creating Front End

Iteration 2 includes the following activities

- Updating Dataset
- Hyper Parameter Tuning
- Modifying Frontend
- Heroku Deployment

3.2 Convolutional Neural Network

Classifier models can be divided into two categories respectively which are generative models based on hand-crafted features and discriminative models based on traditional learning such as support vector machine (SVM), Random Forest (RF), and Convolutional Neural Network (CNN). One difficulty with methods based on hand-crafted features is that they often require the computation of a large number of features to be accurate when used with many traditional machine learning techniques. This can make them slow to compute and expensive memory-wise. More efficient techniques employ lower numbers of features, using dimensionality reduction like PCA (Principle Component Analysis) or feature selection methods, but the reduction in the number of features is often at the cost of reduced accuracy. Brain tumor segmentation employs discriminative models because, unlike generative

modeling approaches, these approaches exploit little prior knowledge on the brains anatomy and instead rely mostly on the extraction of [many] low-level image features, directly modeling the relationship between these features and the label of a given voxel. In our project, we have used the Convolutional Neural Network architecture for Brain tumor Detection. Convolutional neural network processes closely knitted data used for image classification, image processing, face detection, etc. It is a specialized 3D structure with specialized NN analyzing RGB layers of an image. Unlike others, it analyses one image at a time, identifies and extracts important features, and uses them to classify the image. Convolutional Neural Networks (ConvNets) automatically learn mid-level and high-level representations or abstractions from the input training data. The main building block used to construct a CNN architecture is the convolutional layer. It also consists of several other layers, some of which are described as below:

- Input Layer - It takes in the raw pixel value of the input image
- Convolutional Layer - It is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as an image matrix and a filter or kernel to generate a feature map Convolution of an image with different filters can perform operations such as edge detection, blur, and sharpen by applying filters.

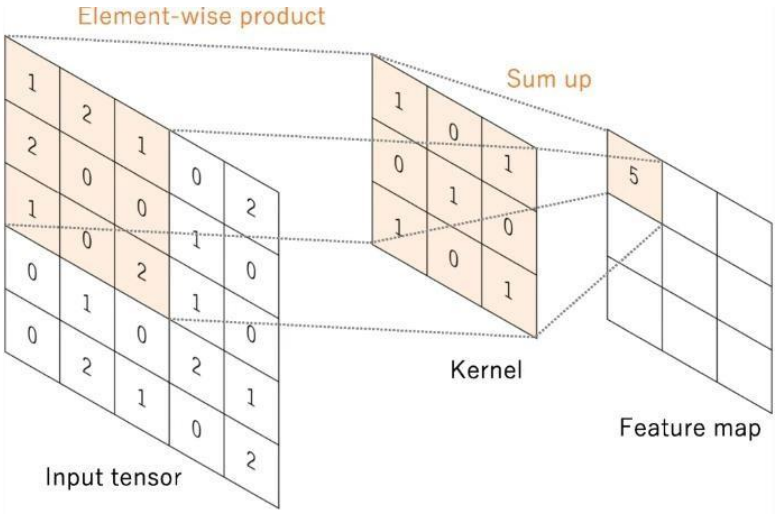

Figure 2 : Convolutional Layer

- Activation Layer - It produces a single output based on the weighted sum of inputs
- Padding - Padding, typically zero padding, is a technique to address this issue, where rows and columns of zeros are added on each side of the input tensor, to fit the center

of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation. Modern CNN architectures usually employ zero padding to retain in-plane dimensions to solving apply more layers. Without zero padding, each successive feature map would get smaller after the convolution operation.

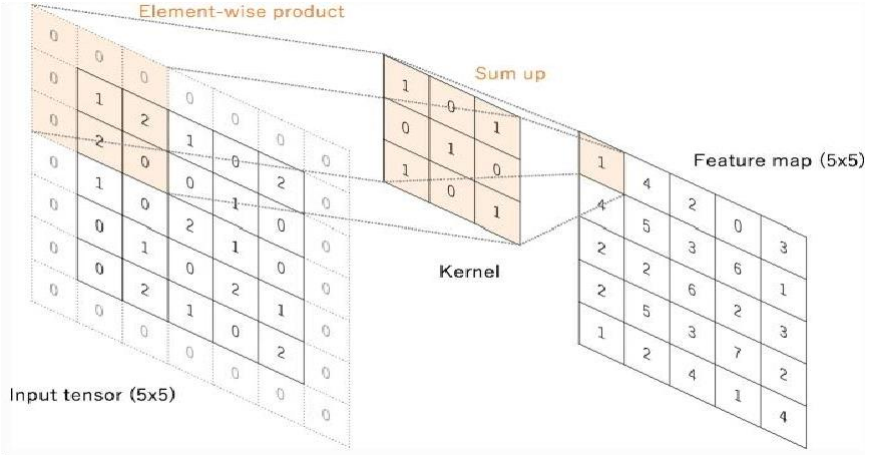

Figure 3 : Padding Layer

- Pooling Layer - The pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling (also called subsampling or downsampling) reduces the dimensionality of each map but retains important information. Spatial Pooling can be of different types:
  - ❖ Max Pooling - taking the largest element in the feature map
  - ❖ Average Pooling - taking the average of elements in the feature map
  - ❖ Sum Pooling - taking the sum of all elements in the feature map

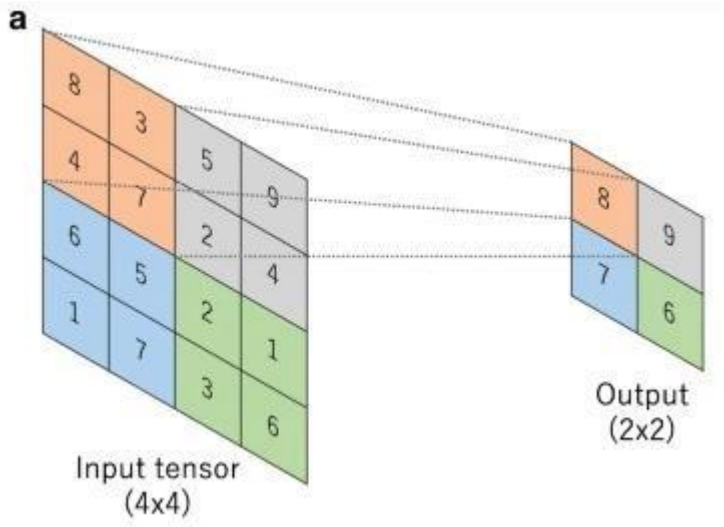

Figure 4 : Max- Pooling Layer

- Fully Connected Layer - In this layer, we flattened our matrix into a vector and feed it

into a fully connected layer like a neural network. The feature map matrix will be converted as a column vector (x1, x2, x3, …). With the fully connected layers, we combined these features to create a model. For classifying input images into various classes based on the training set.

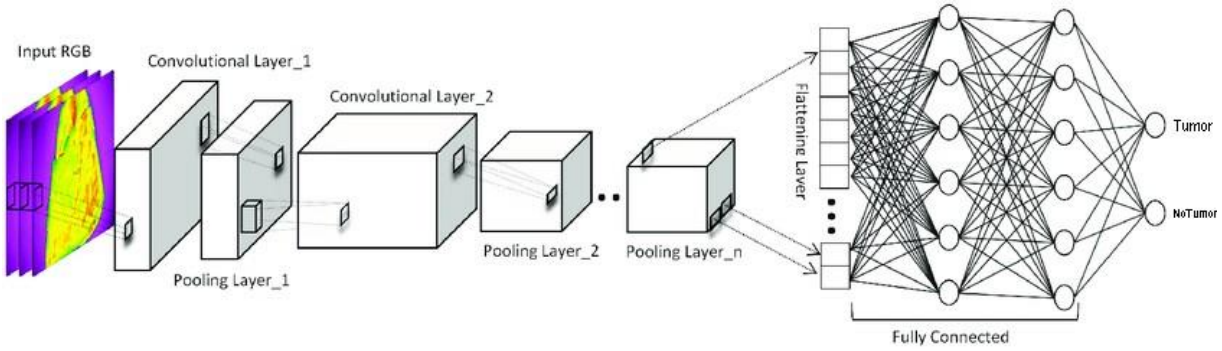

Figure 5 : Convolutional Neural Network

3.3. Datasets

We took our dataset from Kaggle. Our dataset contains two folders, one having tumor images and another having non-tumor images. Both folders contain 1500 images each making the total size of the dataset to be 3000 images. All of the images are in jpg format, and of varying shapes. So we had to resize the images to train the model.

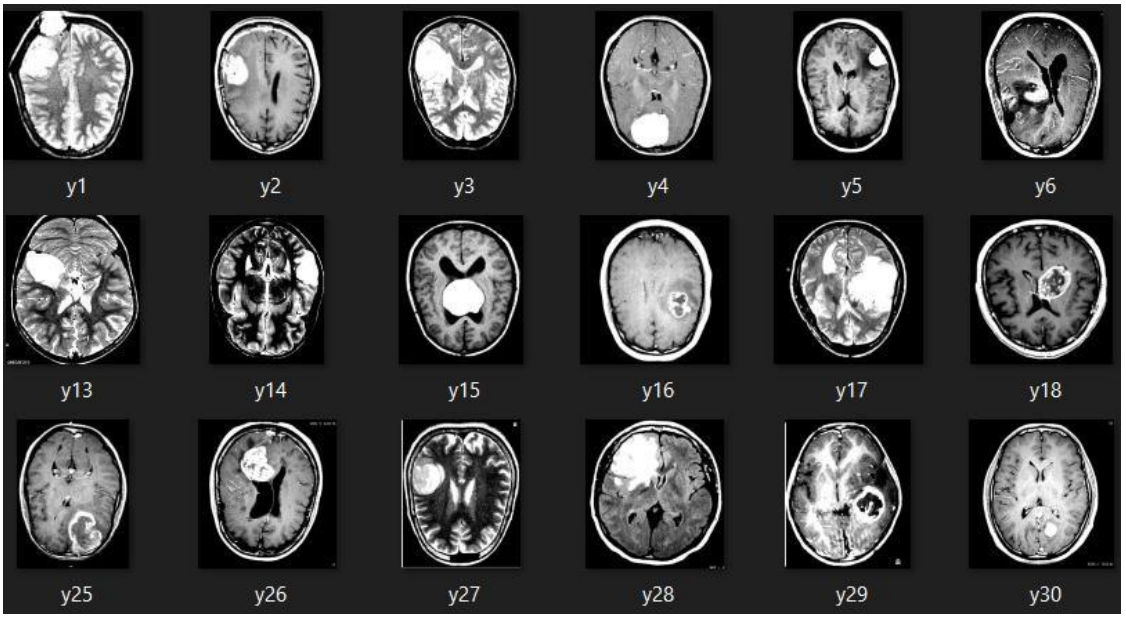

Figure 6 : Tumor Dataset

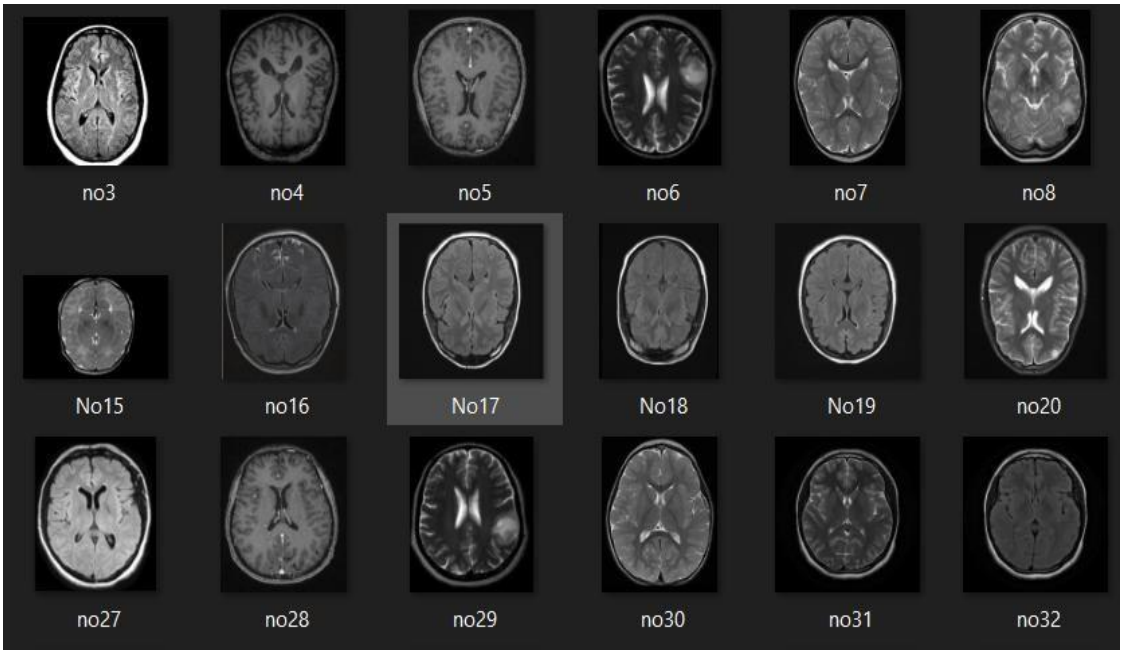

Figure 7 : No-Tumor Dataset

3.4 Tools and Technologies Used

The overall tools used in our project are listed in the table below:

Tools	**Purpose**
Jupyter Notebook	To create a model.
Tensorflow	To build a CNN architecture.
scikit-learn	To split the dataset into testing and training.
NumPy	To read the image as an array.
Matplotlib	To visualize the images and graphs.
Flask	Framework to code web API.
Visual Studio Code	IDE for coding web API and frontend UI
Github	To store code for deployment.
Heroku	To deploy the project on the cloud.

Table 1: Tools and Technology Used

3.5 Team Members and Divided Roles

Team Members	Roles	Responsibility
Puspha Raj Pandeya	Project Manager, Model Developer and Documentation	Develop ML model, document the process and track the progress in the project.
Sandesh Paudel	Model Developer, Backend Developer and Deployment	Assist in model development, create backend system to integrate the model, and Deployment.

Table 2 : Team Members and Divided Roles

4. System Implementation

4.1. Dataflow Diagram

A dataflow is a type of diagram that represents a workflow or process. A dataflow can Also be defined as a step-by-step approach to solving a task. The dataflow diagram for our system is given below.

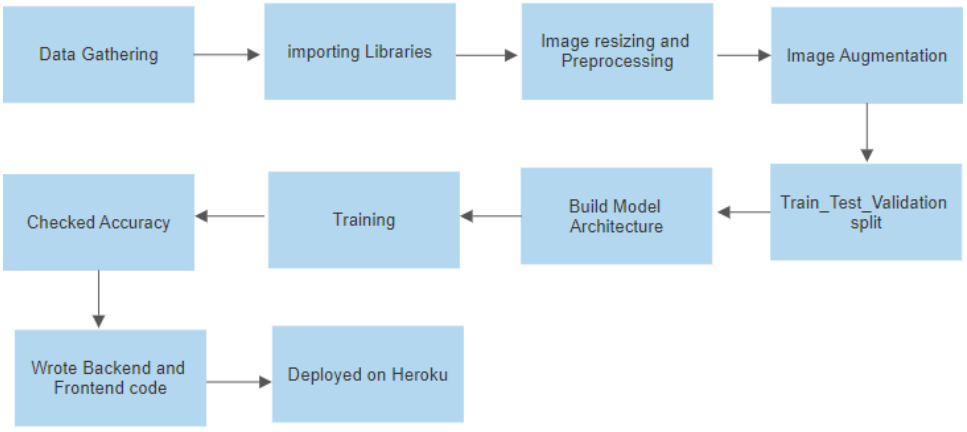

Figure 8 : Dataflow Diagram

4.2. Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The simplified and graphical representation of what our system must do is represented below:

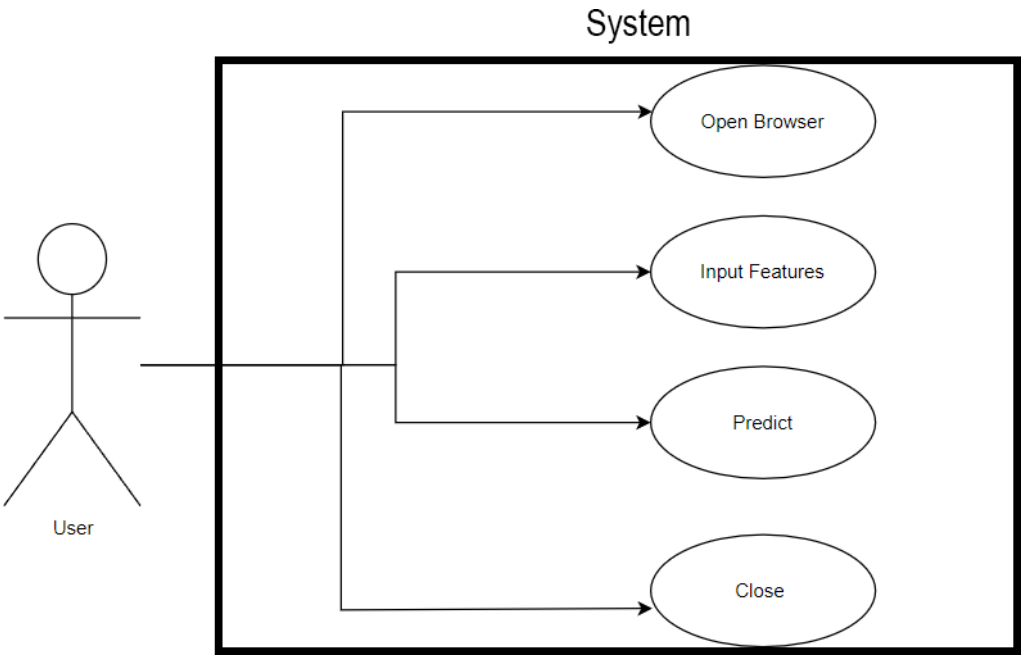

Figure 9 : Use case Diagram

4.3. Model Creation

Different steps were involved while creating a model. Some of the steps are shown below:

1. **Importing required libraries**: A library is a file containing a set of functions that we want to include in your application. Various libraries were used in our project to build the CNN model.

```
import os
import keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('dark_background')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```

Figure 10 : Required Libraries

2. **Image preprocessing**: The images were of various shapes and sizes so we had to resize our images to feed them to our model for training.

```
data = []
paths = []
result = []

for r, d, f in os.walk(r'/content/drive/MyDrive/finalproject/dataset/yes'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[0]]).toarray())
```

Figure 11: Image Processing

3. **Creating model architecture**: Then we created an architecture for our model and performed hyper parameter tuning to get the best accuracy.

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding = 'Same'))
model.add(Conv2D(32, kernel_size=(2, 2),  activation ='relu', padding = 'Same'))


model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'))
model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'))

model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(loss = "categorical_crossentropy", optimizer='Adamax',metrics=['accuracy'])
print(model.summary())
```

Figure 12 : Model architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      416
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 32)      4128
_________________________________________________________________
batch_normalization (BatchNo (None, 128, 128, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        8256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 64)        16448
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 512)               33554944
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 33,585,602
Trainable params: 33,585,410
Non-trainable params: 192
```

Figure 13 : Model Summary

4. **Training the model**: Then we trained our model on the training set and got an accuracy of 97.55%.

```
history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose = 1,validation_data = (x_test, y_test))

Epoch 1/30
58/58 [==============================] - 42s 155ms/step - loss: 7.3799 - accuracy: 0.7522 - val_loss: 0.7975 - val_accuracy: 0.8497
Epoch 2/30
58/58 [==============================] - 8s 131ms/step - loss: 0.3839 - accuracy: 0.8832 - val_loss: 0.2551 - val_accuracy: 0.8981
Epoch 3/30
58/58 [==============================] - 8s 133ms/step - loss: 0.1824 - accuracy: 0.9360 - val_loss: 0.2649 - val_accuracy: 0.9240
Epoch 4/30
58/58 [==============================] - 8s 132ms/step - loss: 0.1286 - accuracy: 0.9520 - val_loss: 0.3543 - val_accuracy: 0.9344
Epoch 5/30
58/58 [==============================] - 8s 131ms/step - loss: 0.1207 - accuracy: 0.9611 - val_loss: 0.3747 - val_accuracy: 0.9326
Epoch 6/30
58/58 [==============================] - 8s 131ms/step - loss: 0.0857 - accuracy: 0.9732 - val_loss: 0.4711 - val_accuracy: 0.9413
Epoch 7/30
58/58 [==============================] - 8s 133ms/step - loss: 0.0674 - accuracy: 0.9823 - val_loss: 0.4988 - val_accuracy: 0.9447
Epoch 8/30
58/58 [==============================] - 8s 132ms/step - loss: 0.0550 - accuracy: 0.9844 - val_loss: 0.4724 - val_accuracy: 0.9499
Epoch 9/30
58/58 [==============================] - 8s 132ms/step - loss: 0.0359 - accuracy: 0.9913 - val_loss: 0.5361 - val_accuracy: 0.9413
Epoch 10/30
58/58 [==============================] - 8s 131ms/step - loss: 0.0340 - accuracy: 0.9905 - val_loss: 0.5292 - val_accuracy: 0.9430
Epoch 11/30
58/58 [==============================] - 8s 133ms/step - loss: 0.0289 - accuracy: 0.9922 - val_loss: 0.5794 - val_accuracy: 0.9292
Epoch 12/30
58/58 [==============================] - 8s 132ms/step - loss: 0.0367 - accuracy: 0.9918 - val_loss: 0.6112 - val_accuracy: 0.9292
Epoch 13/30
58/58 [==============================] - 8s 132ms/step - loss: 0.0337 - accuracy: 0.9905 - val_loss: 0.5436 - val_accuracy: 0.9378
Epoch 14/30
58/58 [==============================] - 8s 131ms/step - loss: 0.0202 - accuracy: 0.9965 - val_loss: 0.5119 - val_accuracy: 0.9447
```

Figure 14 : Training Model

5. **Visualization**: Then we visualized the accuracy and loss graphs and checked how our model has performed.

```
[ ]  plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('Model accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel('Epoch')
     plt.legend(['Test', 'Validation'], loc='upper right')
     plt.show()
```

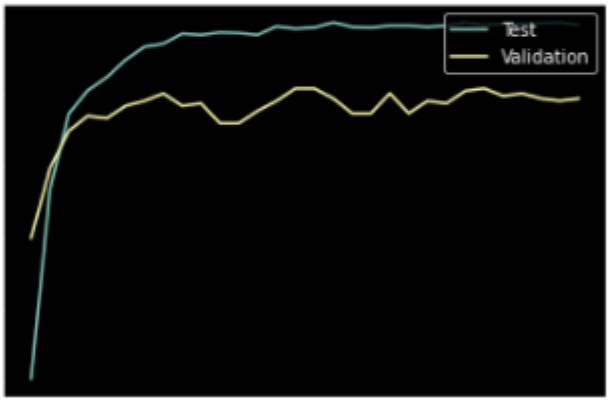

Figure 15 : Accuracy Curve

```
[ ]  plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('Model Loss')
     plt.ylabel('Loss')
     plt.xlabel('Epoch')
     plt.legend(['Test', 'Validation'], loc='upper right')
     plt.show()
```

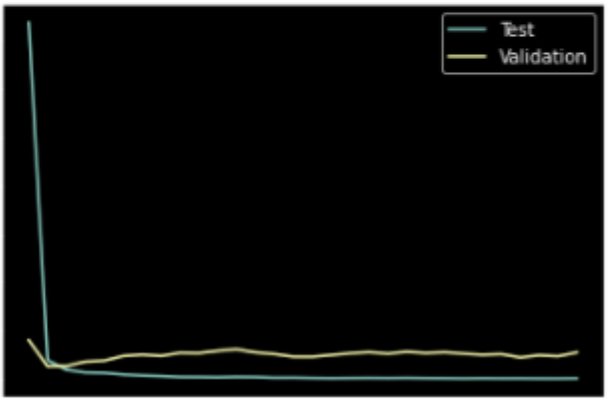

Figure 16 : Loss Curve

6. **Deployment**: Then we build a web API using the flask framework that would take an image as an input, feed it to the model and display the prediction given by the model. The deployment was done on Heroku.

5. Project Task and Time Schedule

The different tasks required to complete this project and estimated time requirements are as follows:

Task Name	Start	Finish	Duration (Days)
Planning	4/15/2021	4/27/2021	12
Analysis	4/28/2021	5/4/2021	6
Project Requirements	5/5/2021	5/14/2021	9
Iteration 1	5/15/2021	6/23/2021	39
Basic Design Diagrams	5/15/2021	5/19/2021	4
Data Gathering	5/20/2021	5/21/2021	1
Image Processing	5/22/2021	5/26/2021	4
Model building	5/27/2021	6/3/2021	7
Created Web API	6/4/2021	6/13/2021	9
Created Front End	6/14/2021	6/17/2021	3
Documentation	5/15/2021	6/23/2021	39
Iteration 2	6/24/2021	7/12/2021	18
Updated Dataset	6/24/2021	6/25/2021	1
Hyper Parameter Tuning	6/26/2021	6/28/2021	2
Modified Frontend	6/28/2021	6/30/2021	2
Heroku Deployment	7/1/2021	7/7/2021	6
Documentation	6/24/2021	7/12/2021	18

Table 3 : Task and Time Schedule

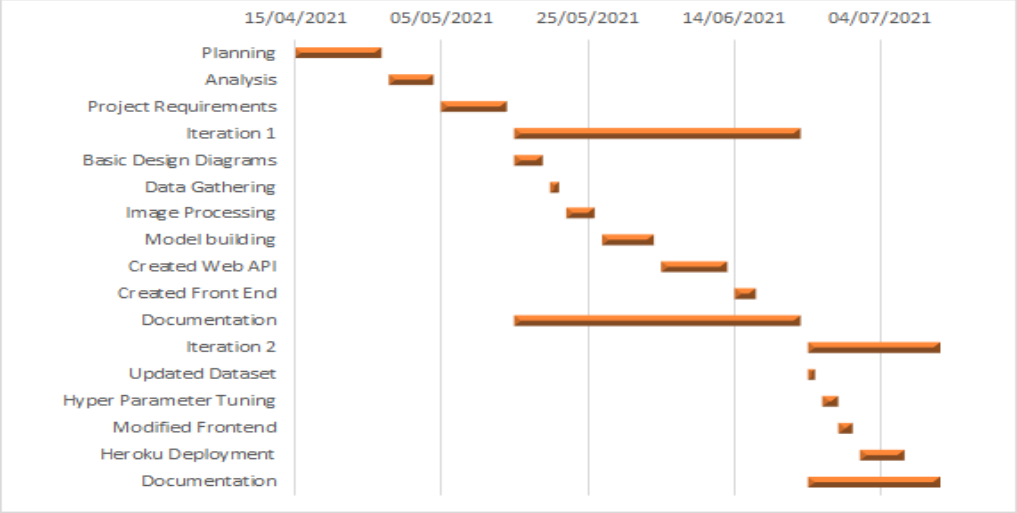

Figure 17 : Gantt chart

6. Conclusion

In Our Project, we have developed open-source software to distinguish normal and abnormal brain MRIs. We have made a GUI, which is user-friendly. Several methodologies exist for brain tumor detection from MRI images, but this method is fast and is freely available for users.

One of the major problems while creating the model was to avoid our model from over fitting. Over fitting happens when your model fits too well to the training set. It then becomes difficult for the model to generalize to new examples that were not in the training set. To avoid over fitting, we took a large dataset and added dropout for regularization while training the model.

As a result of this project, we get an accuracy of 97.5% using CNN and deployed the model into Heroku.

7. Future Work

In the future, further additional functionalities as per requirements will be added which may include charts and graphical comparison. There are so many algorithms available for segmentation and image processing, so by using those algorithms Neurological examination, brain scan, biopsy, perceive the depth, detecting the curves, and inclination for creating maps become easy and accurate.

8. Reference

[1] Ailion, A. S., King, T. Z., Wang, L., Fox, M. E., Mao, H., Morris, R. M., & Crosson, B. (2016). Cerebellar atrophy in adult survivors of childhood cerebellar tumor. *Journal of the International Neuropsychological Society* 1–11. doi:10.1017/s1355617716000138

[2] Ashburner, J., & Friston, K. J. (2000). Voxel-based morphometry—The methods. *NeuroImage*, *11*(6), 805–821. doi:10.1006/nimg.2000.0582.

[3] Aukema, E. J., Caan, M. W., Oudhuis, N., Majoie, C. B., Vos, F. M., Reneman, L., et al. (2009). White matter fractional anisotropy correlates with speed of processing and motor speed in young childhood cancer survivors. *International Journal of Radiation Oncology, Biology, Physics*, *74*(3), 837–843. doi:10.1016/j.ijrobp.2008.08.060.

[4] Brett, M., Leff, A. P., Rorden, C., & Ashburner, J. (2001). Spatial normalization of brain images with focal lesions using cost function masking. *NeuroImage*, *14*(2), 486–500. doi:10.1006/nimg.2001.0845.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[6] https://www.tensorflow.org/guide/keras/sequential_model

[7] https://docs.opencv.org/3.4.14/

Appendix

App Screenshots:

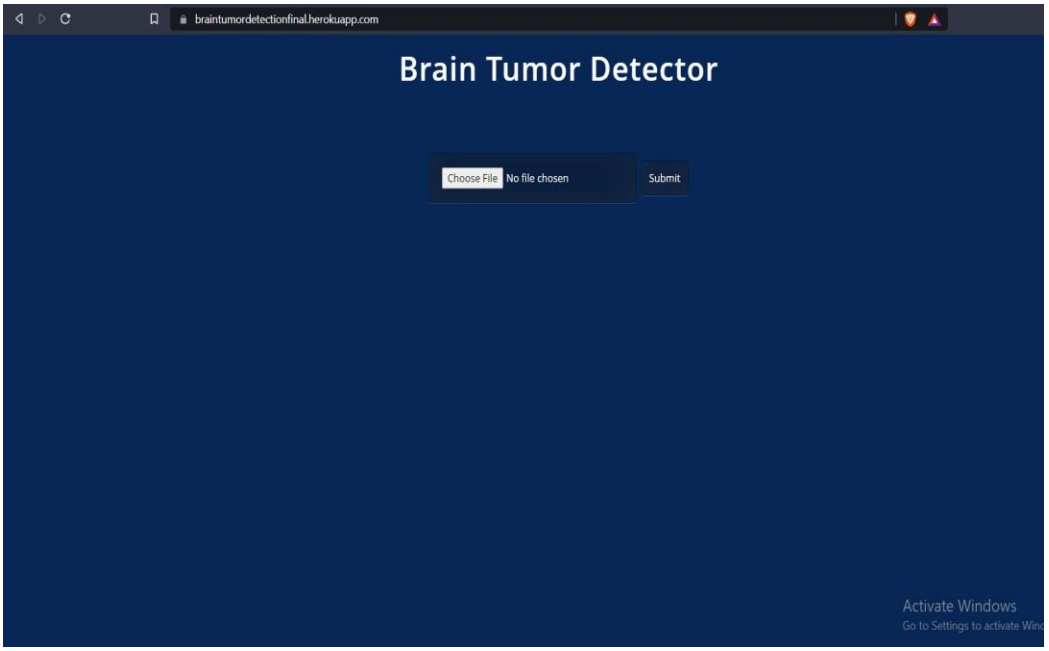

Figure 18: Homepage

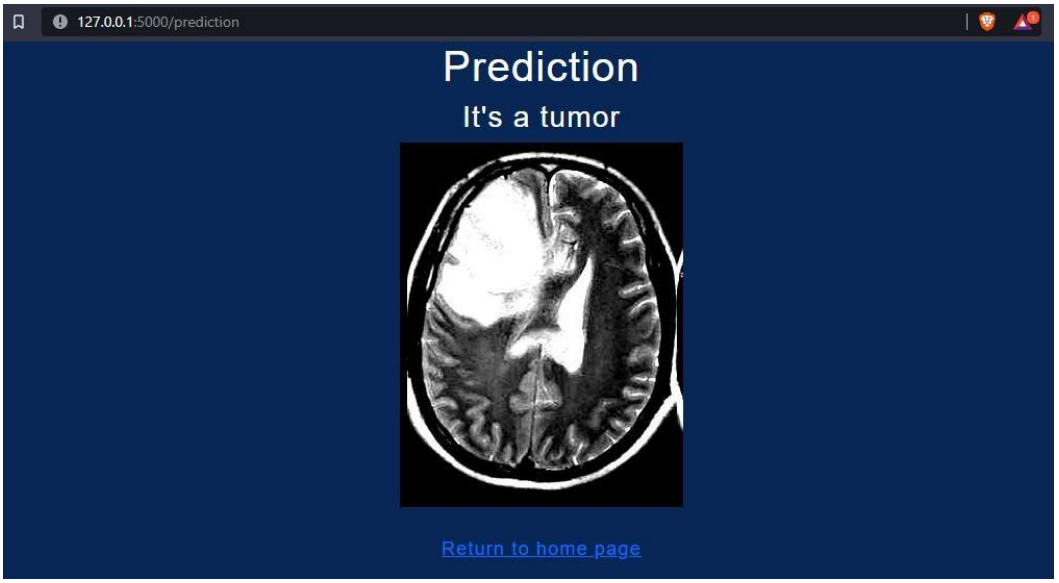

Figure 19 : Prediction page (Tumor Detected)

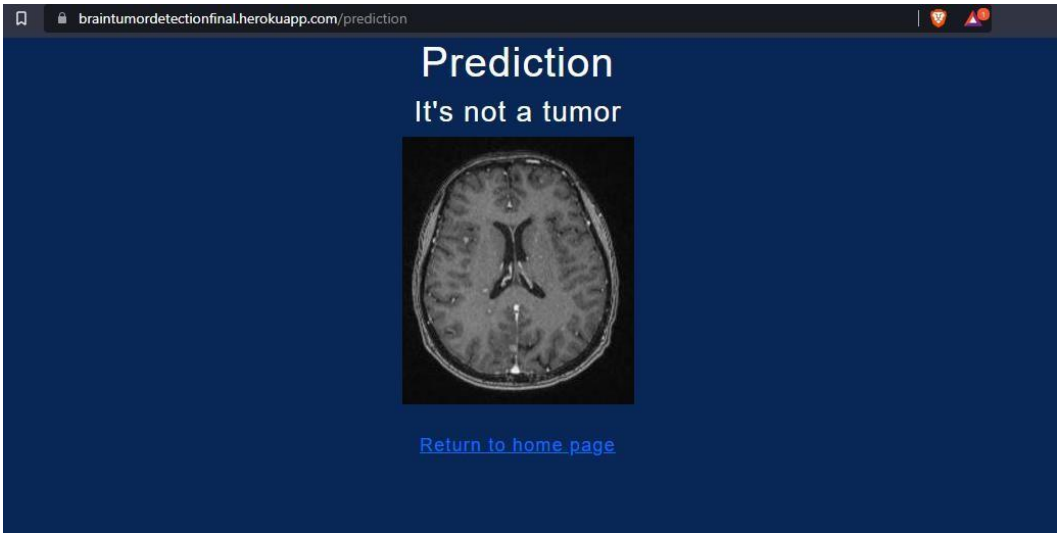

Figure 20 : Prediction page (No Tumor Detected)